Progress Report

Grant No: N00014-94-1-0462

Office of Naval Research

Self-Checking State Machine Realization in CMOS

Reporting Period : July 1 --- December 31, 1994

Dr. P.K. Lala
Principal Investigator
Dr. A. Walker
Co-Principal Investigator
Department of Electrical Engineering
North Carolina A&T State University
Greensboro, NC 27411.

In this reporting period we have accomplished the following:

i.. Developed a procedure for totally self-checking (TSC) checker design for m-out-of-2m codes at the transistor level.

ii. Derived a technique for designing TSC fault-tolerant systems.

i. TSC checker for m-out-of-2m codes.

The *m*-out-of-*2m* (*m*/*2m*) codes are special cases of *m*-out-of-*n* codes which are useful for detecting single bit and unidirectional multibit errors in information bits. The direct mapping of a gate-level TSC checker to its transistor-level equivalent, cannot guarantee TSC property. This is because not all faults at the transistor-level can be modeled as stuck-at faults, which are commonly assumed at the gate level. We have developed an approach for implementing checkers for m/2m codes at the transistor-level which are TSC with respect to the following faults:

a). single stuck-at faults at input and output signal lines;

b). stuck-on and stuck-open transistor faults;

c). bridges between input signal lines;

d). breaks in input signal lines.

e). bridges in source-drain (SD), gate-source (GS) and gate-drain (GD) of transistors.

We first propose a TSC checker for 2/4 code, which is designed by replacing the NAND, NOR gates in a gate-level 2/4 TSC checker with new circuit structures as shown in Fig 1, instead of using traditional CMOS implementation of NAND, NOR gates. Fig.2 shows the transistor-level implementation of TSC checker for 2/4 code.

Theorem: The checker circuit of Fig.2 is TSC for the faults assumed in the last section. For the sake of brevity , the proof of the theorem is not included.

The 2/4 checker is used as a building block for constructing checkers for several other m/2m codes. A general procedure for designing checkers for m/2m codes where m=3,4 5 and 6 is presented .

TSC Checker Design for *m*-out-of-*2m* Codes (*m*=3, 4, 5, 6):

The procedure for TSC checker design consists of the following steps:

Step 1:

Case 1. m is odd (m = 3, 5)

i). Partition inputs $\{x_1 \ldots x_{2m}\}$ into two blocks A and B such that block A has m+1 input variables and block B has m-1 variables. i.e., A=$\{x_1 \ldots x_{m+1}\}$, B=$\{x_{m+2} \ldots x_{2m}\}$.

ii). Connect the input variables in block A to a TSC checker for $\frac{m+1}{2}$-out-of-(m+1) code and identify its output as $(z_1z_2)_A$; For m = 5, invert the input variables in block B and connect these to a TSC 2/4 checker, identify its output as $(z_1z_2)_B$. For m = 3, $(z_1z_2)_B=x_5x_6$.

iii). Connect $(z_1z_2)_A$ and $(z_1z_2)_B$ to the TSC 2-out-of-4 checker.

Case 2. m is even (m = 4, 6)

i). Partition inputs $\{x_1 \ldots x_{2m}\}$ into two blocks A and B, each block has m input variables, i.e., A=$\{x_1 \ldots x_m\}$, B=$\{x_{m+1} \ldots x_{2m}\}$.

ii). Connect input variables in block A to a $\frac{m}{2}$-out-of-m checker, and mark its output as $(z_1z_2)_A$.

iii). Connect input variables in block B to a $\frac{m}{2}$-out-of-m checker, and mark its output as $(z_1z_2)_B$.

iv). Connect $(z_1z_2)_A$ and $(z_1z_2)_B$ to the TSC 2-out-of-4 checker.

Step 2:

i). Partition inputs $\{x_1 \ldots x_{2m}\}$ into two blocks A_I and B_I. For $m = 4, 6$, each block has m inputs, i.e., $A_I = \{x_1 \ldots x_m\}$, $B_I = \{x_{m+1} \ldots x_{2m}\}$. For $m = 3, 5$, block A_I has $m+1$ elements and block B_I has m-1 inputs. i.e., $A_I = \{x_1 \ldots x_{m+1}\}$, $B_I = \{x_{m+2} \ldots x_{2m}\}$.

ii). For $m = 3, 4$, partition block A_I into two blocks a_1 and a_2, each having 2 input variables. i.e., $a_1 = \{x_1\ x_2\}$ and $a_2 = \{x_3\ x_4\}$. For $m = 4$, partition block B_I into two blocks b_1 and b_2 such that $b_1 = \{x_5\ x_6\}$ and $b_2 = \{x_7\ x_8\}$. Let block $A_{II} = \{a_2\ B_I\}$ and $B_{II} = a_1$ for $m = 3$, and let block $A_{II} = \{a_2\ b_2\}$ and $B_{II} = \{a_1\ b_1\}$ for $m = 4$. Connect blocks A_I and B_I to a checker block I designed by Step 1, identify its output as $(z_1z_2)_I$. Connect blocks A_{II} and B_{II} to a checker block II designed by Step 1, identify its output as $(z_1z_2)_{II}$.

iii). For $m = 5$, partition block A_I into three blocks a_1, a_2 and a_3 such that $a_1 = \{x_1\ x_2\}$, $a_2 = \{x_3\ x_4\}$ and $a_3 = \{x_5\ x_6\}$, and partition block B_I into two blocks b_1 and b_2 such that $b_1 = \{x_7\ x_8\}$ and $b_2 = \{x_9\ x_{10}\}$. For $m = 6$, partition block A_I into two blocks a_1 and a_2 such that $a_1 = \{x_1\ x_2\ x_3\}$ and $a_2 = \{x_4\ x_5\ x_6\}$, and partition block B_I into two blocks b_1 and b_2 such that $b_1 = \{x_7\ x_8\ x_9\}$ and $b_2 = \{x_{10}\ x_{11}\ x_{12}\}$. For $m = 5$, let block $A_{II} = \{a_2\ a_3\ b_2\}$, $B_{II} = \{a_1\ b_1\}$, $A_{III} = \{a_3\ b_1\ b_2\}$ and $B_{III} = \{a_1\ a_2\}$. For $m = 6$, let block $A_{II} = \{a_2\ b_2\}$, $B_{II} = \{a_1\ b_1\}$, $A_{III} = \{a_1\ b_2\}$ and $B_{III} = \{a_2\ b_1\}$. Connect blocks A_I and B_I to a checker block I designed by Step 1, identify its output as $(z_1z_2)_I$. Connect blocks A_{II} and B_{II} to a checker block II designed by Step 1, identify its output as $(z_1z_2)_{II}$. Connect blocks A_{III} and B_{III} to a checker block III designed by Step 1, identify its output as $(z_1z_2)_{III}$.

iv). For $m = 3, 4$, connect $(z_1z_2)_I$ and $(z_1z_2)_{II}$ to a TSC Two-rail checker (TRC) to produce the checker's final output.

v). For $m = 5, 6$, connect $(z_1z_2)_{III}$ and $(z_2z_1)_I$ to a TSC TRC that produces outputs A_1 and A_2. Connect $(z_1z_2)_I$ and $(z_1z_2)_{II}$ to a TSC TRC to produce outputs B_1 and B_2.

Finally, connect A_1, A_2, B_1 and B_2 to a TSC TRC to produce the final output. The symbol of the TSC TRC and its input-output pattern are shown in Figure 3. In the following discussion, n represents the number of 1s at the checker's input, n_A is the number of 1s at input block A and n_B is the number of 1s at input block B and so on.

TSC Checker Design for 3/6 and 4/8 codes

We illustrate the above procedure by designing TSC checkers for 3/6 code (m=3), and 4/8 code (m=4). The checker blocks designed by following step 1 of the general procedure, are shown in Figure 4(a) and (b) for $m = 3$, and $m = 4$ respectively.

The two separate partitions on the input variables generated from step 2 are:

For $m = 3$,

i) $A_I = \{x_1\ x_2\ x_3\ x_4\}$ and $B_I = \{x_5\ x_6\}$;

ii) $A_{II} = \{x_3\ x_4\ x_5\ x_6\}$ and $B_{II} = \{x_1\ x_2\}$.

For $m = 4$,

i). $A_I = \{x_1\ x_2\ x_3\ x_4\}$ and $B_I = \{x_5\ x_6\ x_7\ x_8\}$;

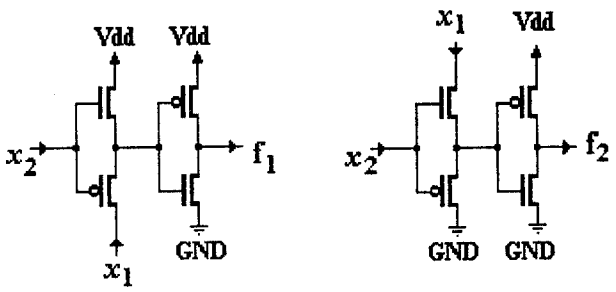
ii). $A_{II} = \{x_3\ x_4\ x_7\ x_8\}$ and $B_{II} = \{x_1\ x_2\ x_5\ x_6\}$.

Inputs belonging to a partition are connected to checker blocks as shown in Fig. 5(a) and 5(b) respectively. The outputs of the checker blocks corresponding to partitions I and II are identified by $(z_1z_2)_I$ and $(z_1z_2)_{II}$ respectively. These outputs feed a TSC TRC to produce the final output of the 3/6 and 4/8 checker. As shown in Tables 1 and 2 both of these checkers satisfy the code-disjoint property.

The TSC checkers for 5/10 code and 6/12 code are designed in a similar manner, and are shown in Fig. 6 and Fig.7 respectively.

ii. TSC Fault tolerant System Design:

This work has resulted in a paper which has been accepted for conference presentation and publication in the Proceedings of the Second International Conference on Reliability and Quality in Design (RQD'95 Proceedings). A copy of the paper is attached.

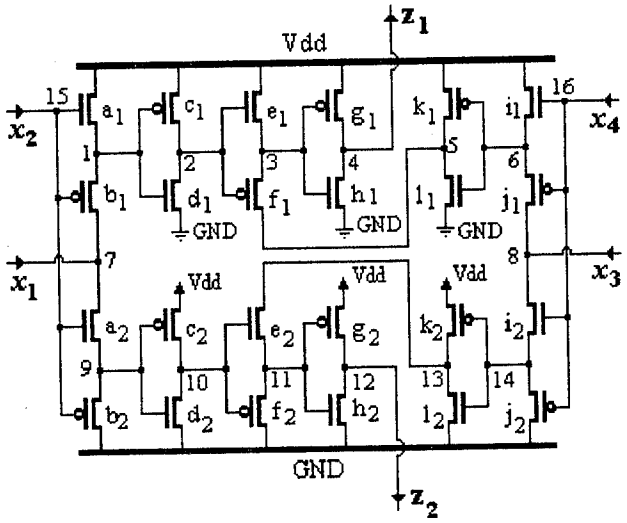


(a) (b)

$x_1\, x_2$	$f_1 = \overline{x_1 + x_2}$	$f_2 = \overline{x_1 \cdot x_2}$
0 0	1	1
0 1	0	1
1 0	0	1
1 1	0	0

Fig. 1. CMOS implementation of NAND, NOR functions

***(a).* NOR circuit. *(b).* NAND circuit.**

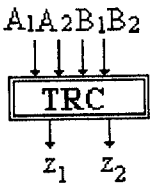


non-code word $x_1 x_2 x_3 x_4$	output $z_1 z_2$
0 0 0 0	0 0
0 0 0 1	0 0
0 0 1 0	0 0
0 1 0 0	0 0
1 0 0 0	0 0
0 1 1 1	1 1
1 0 1 1	1 1
1 1 0 1	1 1
1 1 1 0	1 1
1 1 1 1	1 1

codeword $x_1 x_2 x_3 x_4$	output $z_1 z_2$
0 0 1 1	0 1
0 1 0 1	1 0
0 1 1 0	1 0
1 0 0 1	1 0
1 0 1 0	1 0
1 1 0 0	0 1

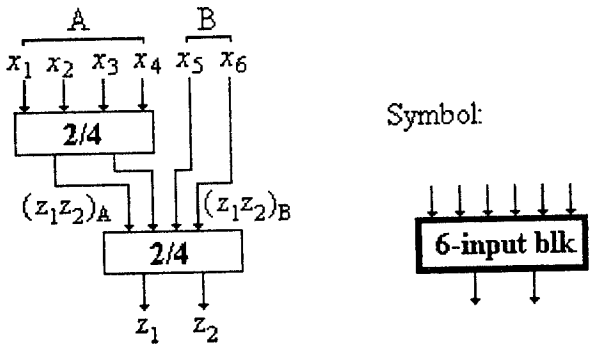
Fig. 2. TSC checker for 2-out-of-4 code

TSC TRC Symbol:

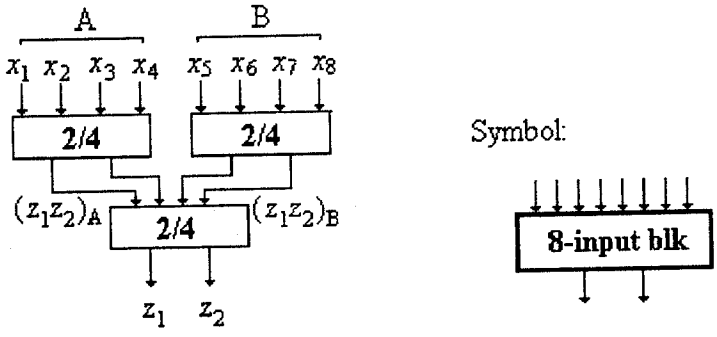


TRC	$A_1A_2B_1B_2$	z_1z_2
codewords	0 1 0 1	1 0
	0 1 1 0	0 1
	1 0 0 1	0 1
	1 0 1 0	1 0
non-code words	0 0 0 0	1 1
	0 0 0 1	1 1
	0 0 1 0	1 1
	0 0 1 1	1 1
	0 1 0 0	1 1
	0 1 1 1	0 0
	1 0 0 0	0 0
	1 0 1 1	1 1
	1 1 0 0	0 0
	1 1 0 1	0 0
	1 1 1 0	0 0
	1 1 1 1	0 0

Figure 3. TSC two-rail checker (TRC)

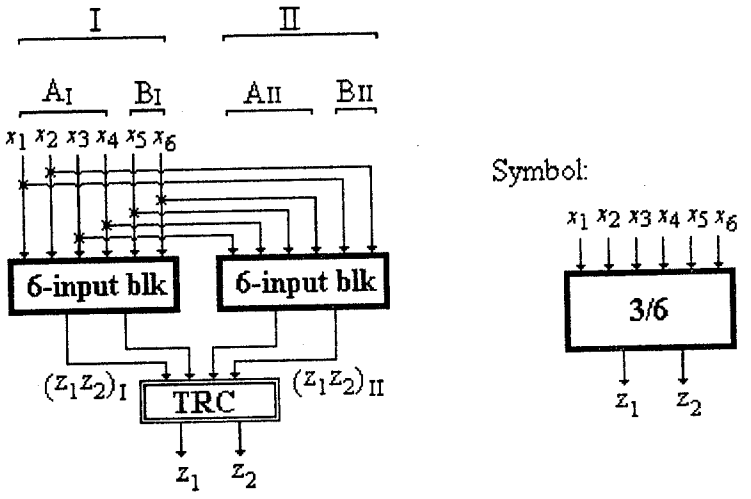


(a)

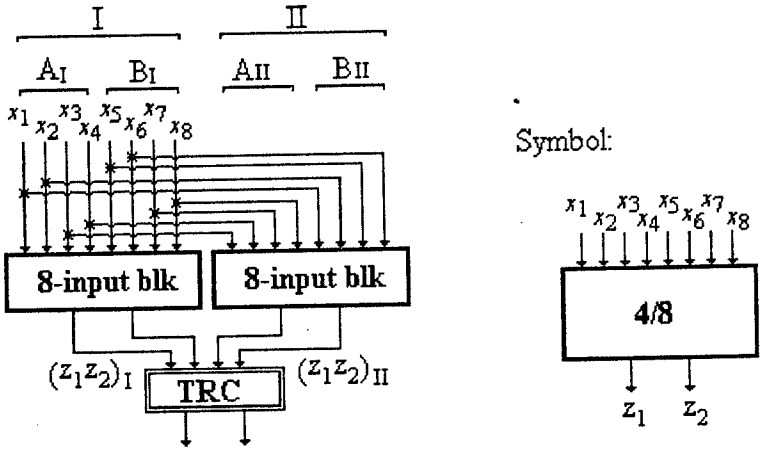


(b)

Figure 4. (a) 6-input checker block; (b) 8-input checker block



(a)



(b)

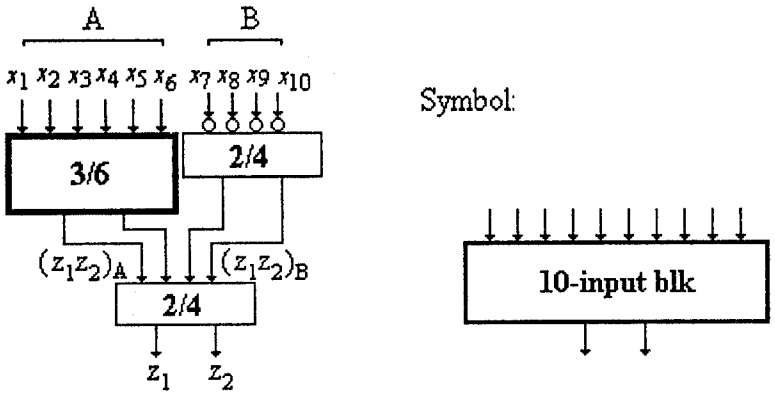
Figure 5. (a) TSC checker for 3/6; (b) TSC checker for 4/8 code.

Table 1. Code disjoint property of the 3/6 checker.

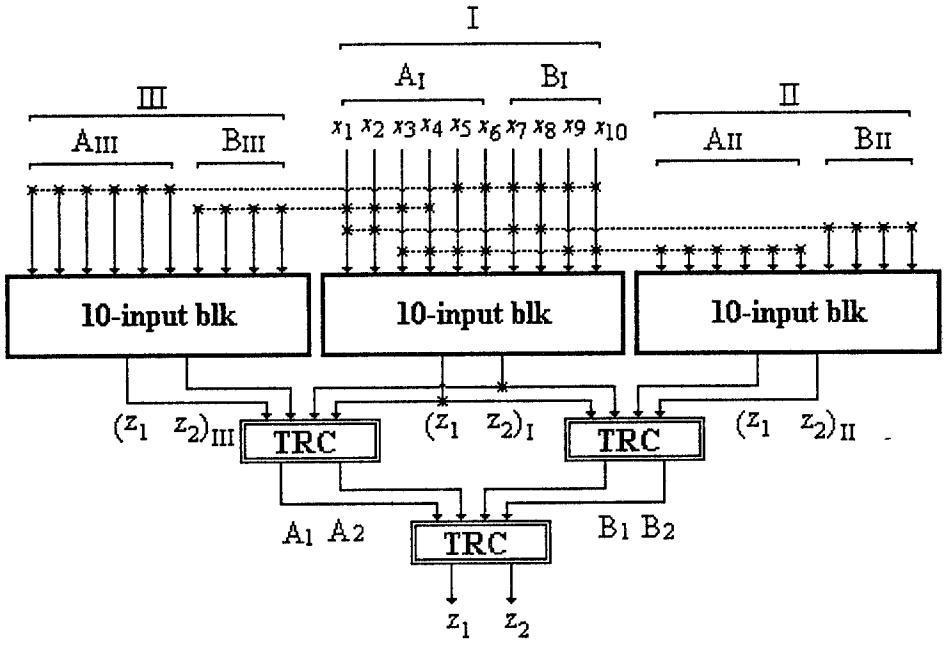
3/6 checker inputs				TRC inputs		3/6 checker outputs
	n	$n_{A_I}\, n_{B_I}$	$n_{A_{II}}\, n_{B_{II}}$	$(z_1z_2)_I$	$(z_1z_2)_{II}$	z_1z_2
codewords	3	1 2	3 0	01	01	10
			2 1	01	10	01
		2 1	3 0, or 1 2	10	01	01
			2 1	10	10	10
		3 0	2 1	01	10	01
			1 2	01	01	10
non-code words	0	00	00	00	00	11
	1	0 1, or 1 0	1 0, or 0 1	00	00	11
	2	**0 2**	**2 0**	**01**	**00**	**11**
		2 0	2 0, or 1 1	00	00	
			0 2	00	01	
		1 1	2 0, or 1 1	00	00	
	4	**4 0**	**2 2**	**01**	**11**	**00**
			3 1, or 2 2	11	11	
		3 1	3 1, or 2 2	11	11	
		2 2	4 0	11	01	
	5	3 2	3 2, or 4 1	11	11	00
		4 1	3 2	11	11	
	6	4 2	4 2	11	11	00

Table 2. Code disjoint property of the TSC checker for 4/8 code.

4/8 checker inputs				TRC inputs		4/8 checker outputs
	n	$n_{A_I}\, n_{B_I}$	$n_{A_{II}}\, n_{B_{II}}$	$(z_1z_2)_I$	$(z_1z_2)_{II}$	z_1z_2
codewords	4	0 4, or 4 0	2 2	01	10	01
		1 3, or 3 1	2 2	01	10	01
			1 3, or 3 1	01	01	10
		2 2	2 2	10	10	10
			0 4, 4 0, 1 3, or 3 1	10	01	01
non-code words	0	0 0	0 0	00	00	11
	1	0 1, or 1 0	0 1, or 1 0	00	00	11
	2	0 2, 2 0, or 1 1	0 2, 2 0, or 1 1	00	00	11
	3	**0 3, or 3 0**	**1 2, or 2 1**	**01**	**00**	**11**
		1 2, or 2 1	1 2, or 2 1	00	00	
			0 3, or 3 0	00	01	
	5	**1 4, or 4 1**	**2 3, or 3 2**	**01**	**11**	**00**
		2 3, or 3 2	2 3, or 3 2	11	11	
			1 4, or 4 1	11	01	
	6	2 4, or 4 2	2 4, 4 2, or 3 3	11	11	00
	7	3 4, or 4 3	3 4, or 4 3	11	11	00
	8	4 4	4 4	11	11	00

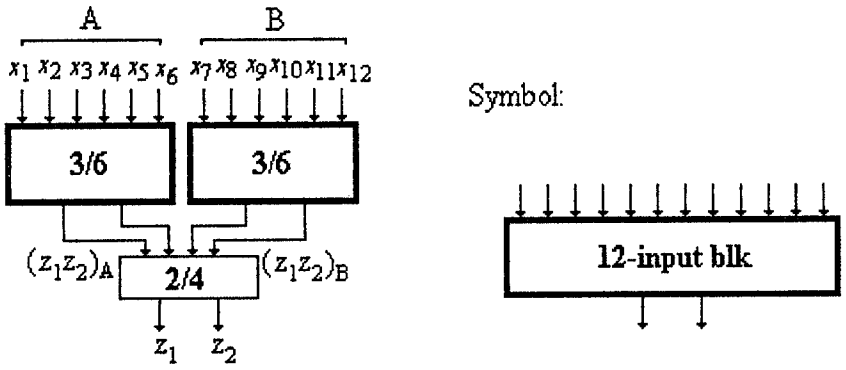


(a)

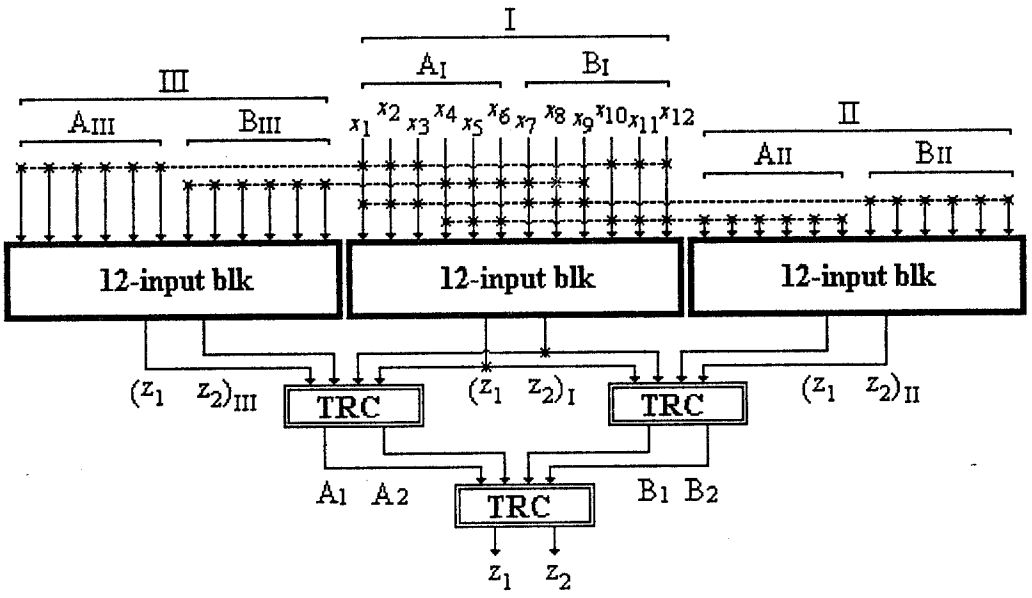


(b)

Fig.6 Totally Self-Checking checker for 5-out-of-10 code



(a)



(b)

Fig. 7 Totally Self-Checking checker for 6-out-of-12 code

TOTALLY SELF-CHECKING FAULT-TOLERANT SYSTEM DESIGN

P.K.Lala and F. S. Vainstein
Dept. of Electrical Engineering
North Carolina A&T State University
Greensboro, NC 27411, USA.

Key Words: Triple Modular redundancy, Self-checking Circuits, Retry

Abstract

A scheme for designing fault tolerant systems which are also totally self-checking for all single faults, is presented in this paper. The system will provide correct output in the presence of a single faulty element and identify the element as well. The scheme also allows distinction between a permanent fault and a transient/intermittent fault.

1. Introduction

One of the established methods of designing a reliable system from less reliable components is the TMR(Triple Modular Redundancy) technique[1]. The output of a TMR system is the majority of three identical components. Thus, such a system can tolerate errors in any one component. The major drawback of the TMR system is that if two modules fail or the voter has a fault which cannot be masked, then the system produces erroneous outputs without giving any indication of failure. Two approaches have been proposed to overcome this problem[2,3]. Both approaches incoporate error-checking circuits to detect erroneous outputs. However, both approaches suffer from the disadvantage that the additional circuitry is not self-checking. Recently, another approach has been proposed to implement totally self-checking TMR fault-tolerant systems[4]. This approach allows detection of both unmasked and masked faults; however, no distinction is made between a permanent and a transient/intermittent fault. Moreover, the circuit overhead is high.

2. Fault tolerant implementation

We propose a new scheme for fault-tolerant system design which also makes the system totally self-checking; the concept of self-checking design has been discussed in [5].In the proposed scheme, the simplex(non-redundant) circuit is replaced by three identical copies X,Y and Z, as shown in Fig.1. As in the TMR system, all three copies receive the same input. The output of each module is compared with the outputs of the remaining two. The outputs of the comparators are identified as a,b and c. If a=0, the outputs of modules X and Y match, whereas a=1 indicates a mismatch between the outputs of the two modules. Similarly, b and c indicate the compared values of modules X/Z and Y/Z respectively. It would be clear that if a module produces faulty output, the outputs of two comparators will be at 1 i.e the outputs of the comparators will form a 2-out-of-3 code. On the other hand, if one comparator is faulty the values of a,b and c will constitute a 1-out-of-3 code. Table 1 shows how a faulty component can be identified from the values of a,b, c. If one of the comparators is faulty, the single module fault assumption is no longer valid, hence no corrective action can be taken. Also, if abc=111, at least two modules are faulty, the corrective action is not activated.

The function of the decision and correction logic in Fig.1 is to reconfigure the system so that the system output is derived from a fault-free module. As mentioned previously, depending on the outputs of the comparators a, b and c, one module is selected to provide the correct output. The decision logic consists of an encoder, a totally self-checking checker and circuitry for enabling the tri-state buffers. The encoder accepts the outputs of the comparators and converts them into a 2-out-of-4 code as shown in Table 2. A totally self-checking 2-out-of-4 checker is placed at the output of the encoder circuit to check the validity of the codeword.

Finally, the output of the system is derived by enabling one of the buffers as indicated in Table 1. If a comparator is faulty, or two or more modules are faulty i.e. lmnp = -00- or -11-, a flag is generated and all the modules are disconnected from the output bus, thus preventing the propagation of erroneous

information. If there is no fault, the output of the enable circuit will form a 1-out-of-4 code. A checker circuit, which is totally self-checking for single and unidirectional multiple errors, is placed at the output of the enable circuit. The checker will produce a 1-out-of-2 code if it receives a 1-out-of-4 code, and if there is no fault in the circuit itself. If the checker produces 00 or 11 output, the system needs repair.

The function of the retry circuit (Fig.2) is to distinguish between a permanent fault and a transient/intermittent fault. It is assumed that the duration of a transient fault is less than two clock periods. Once a fault is detected i. e. abc $\neq$ 000, it is checked whether the fault is of transient or permanent nature. This is accomplished by clocking in the values of abc in a 3-bit register. If abc $\neq$ 000, there is a faulty component, the **error** signal will go to 1, and the contents of the register will feed the AND gate inputs via the multiplexers. If the next set of values of abc is exactly the same as that stored in the register, the corresponding fault is assumed to be of permanent nature, whereas abc = 000 will indicate that the fault is of transient/intermittent nature. If a fault is found to be permanent, it can be diagnosed to a replaceable component which is identified by the contents of the 3-bit register. For example, if module X has a permanent fault, the contents of the register for two consecutive pulses will be 110 (as indicated in Table 1). The retry circuit can be tested off-line for single faults

3. Conclusion

A scheme for improving the reliabilty of digital systems by incorporating fault tolerance and self-checking concepts is presented in this paper. The major advantage of this scheme is that it will not only provide correct output in the presence of a single faulty element, comparator or functional module, but will identify the faulty element as well. In addition, the scheme allows distinction between a permanent and a transient fault in an element; The system is implemented in a modular fashion, and each module is protected by a self-checking checker. In the event of a fault, the system will indicate its presence on-line. If there is a fault combination whose effect cannot be corrected, the output bus is disconnected from the system, thus preventing the propagation of erroneous information to other systems which may be connected to the faulty system.

4. References

[1] J. Von Neumann " Probabilistic logics and the syntheses of reliable organisms from unreliable components". Automata Studies, Princeton Univ. Press,1958, pp.43-98.

[2] C.V.Ramamoorthy and Y.W.Han "Reliability analysis of systems with concurrent detection" IEEE Trans.Comput., vol.C-24, Sept.1975, pp.868-878.

[3] B.Courtois "On balancing safety and reliability of hybrid and B_i-duplexed systems"Proc. FTCS-6, 1976, pp.52-57.

[4] N. Gaitanis "The design of totally self-checking TMR fault-tolerant systems" IEEE Trans. Comput., vol.37, Nov.1988, pp.1450-1454.

[5] P.K.Lala Fault tolerant and Fault Testable Hardware Design, Prentice Hall, 1985.

Acknowledgement: This work was supported in part by the Office of Naval Research under contract N00014-94-1-0462.

a b c	Faulty component	Correct output source
0 0 0	none	module X (or Y or Z)
0 0 1	comparator c	none
0 0 1	comparator c	none
0 1 0	comparator b	none
0 1 1	module Z	module Y (or X)
1 0 0	comparator a	none
1 0 1	module Y	module Z
1 1 0	module X	module Z

Table 1 Reconfiguration of faulty modules

a b c	l m n p
0 0 0	0 0 1 1
0 1 1	1 1 0 0
1 0 1	0 1 0 1
1 1 0	1 0 1 0
1 1 1	1 0 0 1

Table 2 Encoding of 3-bit binary patterns using 2-out-of-4 code

p q r s	z_1 z_2
0 0 0 0	0 0
0 0 0 1	1 0
0 0 1 0	1 0
0 0 1 1	0 0
0 1 0 0	0 1
0 1 0 1	0 0
0 1 1 0	0 0
0 1 1 1	0 0
1 0 0 0	0 1
1 0 0 1	0 0
1 0 1 0	0 0
1 0 1 1	0 0
1 1 0 0	0 0
1 1 0 1	0 0
1 1 1 0	0 0
1 1 1 1	0 0

Table 3 1-out-of-4 to 1-out-of-2 conversion.

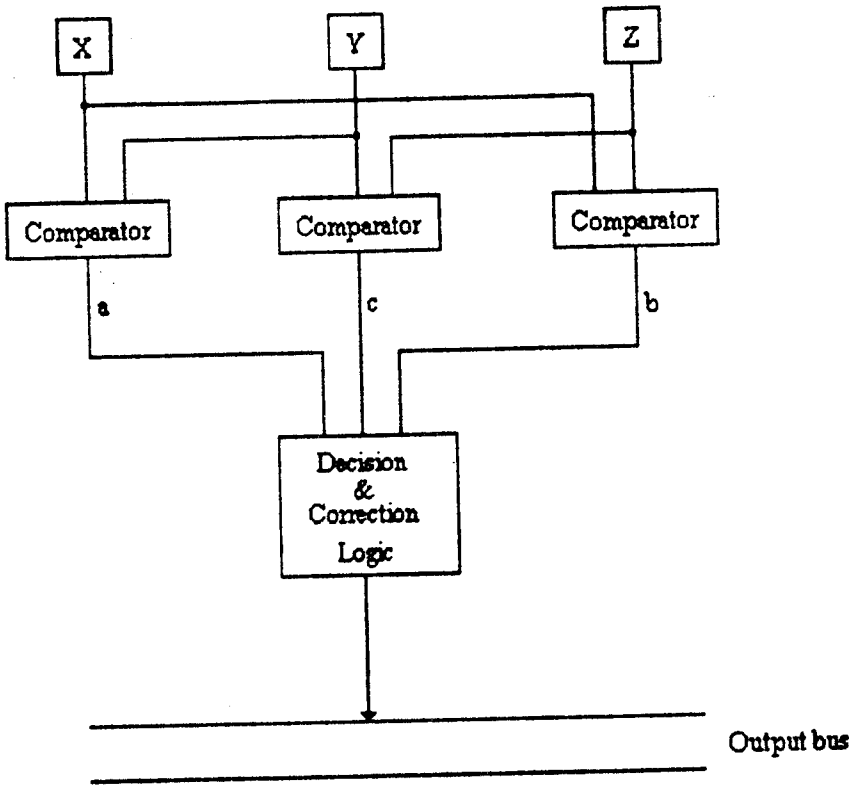


Fig. 1 Fault tolerant system block diagram

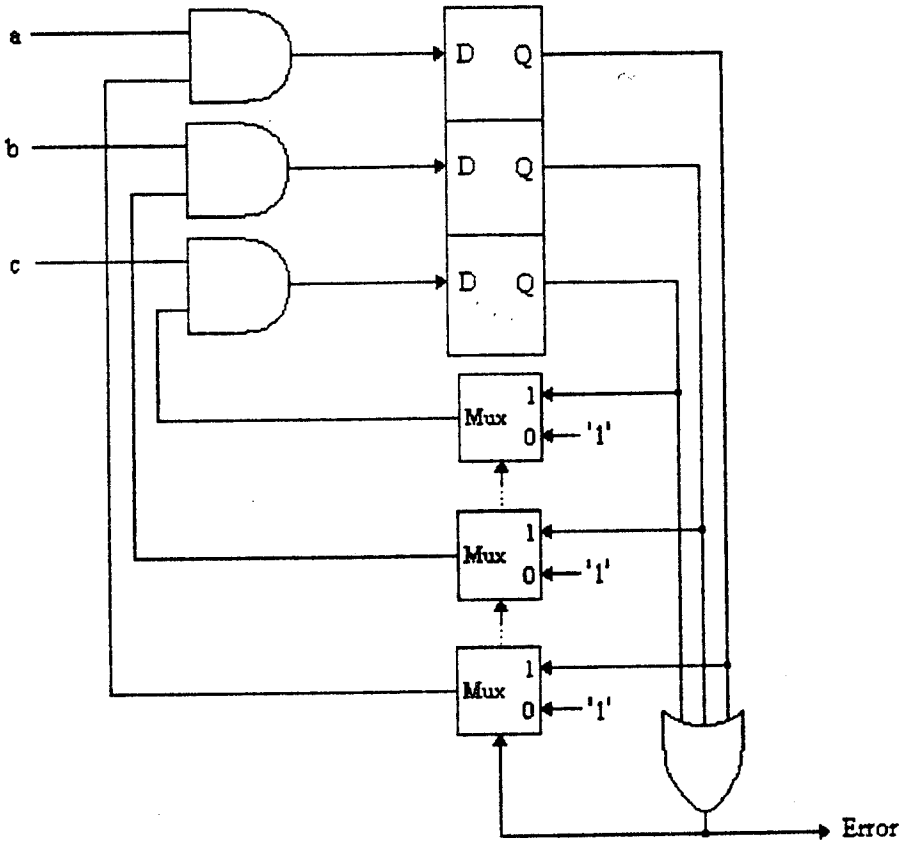


Fig. 2 Retry Circuit